Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (9): 1711-1718, 2009
ISSN 1812-5654

An Efficient Δ-Causal Algorithm for Real-Time Distributed Systems

[1]S.E. Pomares Hernandez, [1]E. Lopez Dominguez, [1]G. Rodriguez Gomez and [2]J. Fanchon
[1]Department of Computer Science, National Institute of Astrophysics, Optics and Electronics,
Luis Enrique Erro No. 1, 72840, Tonantzintla, Puebla, Mexico
[2]Université de Toulouse, LAAS-CNRS, 7 Ave. Colonel Roche, 31077 Toulouse Cedex, France

Abstract: The study presents an efficient Δ-causal algorithm for the transmission of real-time continuous media (e.g., audio and video) in distributed systems. The Δ-causal algorithm is oriented to be used in real-time collaborative applications, such as Teleconferece and Teleimmersion. The Δ-causal algorithm ensures causal delivery of messages with time constraints in partial-reliable and asynchronous networks without using global references. To achieve this, the algorithm introduces an original Forward Error Correction (FEC) mechanism and a method to calculate the message lifetime based on relative time points. One interesting aspect of the FEC mechanism is that the redundant data sent is dynamically adapted according to the behavior of the system. Finally, it is shown that the Δ-causal algorithm is efficient in terms of the overhead (causal history) attached per message.

Key words: Real-time distributed systems, group communication, Δ-causal order

INTRODUCTION

Many studies concerning protocols of causal delivery of messages (Pomares *et al*., 2004; Lopez *et al*., 2008; Plesca *et al*., 2006; Nishimura *et al*., 2005) exist. Most of the causal algorithms assume a reliable transmission without an associated lifetime per message. These studies are not suitable for real-time systems since these systems intrinsically have time constraints that are not considered by the previous studies mentioned. Real-time systems are characterized by two main attributes. First, the information has an associated lifetime that establishes the period of time in which the information (messages) can be received; a message that arrives after its lifetime is useless and consequently, discarded. The second attribute establishes that there is no time for retransmission when messages are lost. In order not to greatly affect the quality of service, a forward recovery scheme is preferable over a backward recovery scheme (Perkins, 2003).

The causal order with time constraints has earlier been addressed by Baldoni *et al*. (1998) and it is called Δ-causal order. Baldoni *et al*. (1998) ensured Δ-causal order when messages are lost by using a global clock. The algorithm that is proposed in this article ensures Δ-causal order in unreliable networks while avoiding the use of global references. To achieve this, an original FEC mechanism as well as a method to calculate, in a distributed manner, the lifetime per message is proposed. The FEC mechanism ensures that causal order delivery is accomplished even in the presence of lost messages. The lifetime in this study is calculated based on relative time points. The study presented here is mainly intended for the transmission of real-time data, such as audio and video.

The FEC mechanism, as well as the distributed lifetime method to extend the minimal causal broadcast algorithm presented by Pomares *et al*. (2004), is applied. This minimal algorithm only sends control information about messages with an Immediate Dependency Relation (IDR). Messages related by an IDR have a causal distance (Definition 4) of one (i.e., no intermediate causal message exists between them). In order to support delays and loss of messages, redundancy is introduced on the control information by increasing the causal distance. One interesting aspect of this FEC mechanism is that the redundancy is dynamically adapted according to the behavior of the system.

The most important study that tackles the problem of causality and time constraints was presented by Baldoni *et al*. (1998). He addresses the problem of causality and time constraints by introducing the Δ-causal order. Informally, the Δ-causal order says that a message m is Δ-causally-related to another message m' if m causally precedes m' and arrives before its deadline. Baldoni ensures message Δ-causal order by using a

Corresponding Author: Saul E. Pomares Hernandez, Department of Computer Science, National Institute of Astrophysics, Optics and Electronics, Luis Enrique Erro No. 1, 72840 Tonantzintla, Puebla, Mexico
Tel: +55 (222) 266.31.00/8227 Fax: +55 (222) 266.31.52

reference global clock. More specifically, by using a global clock, Baldoni determines if a message accomplishes the causal order and the real-time delivery constraints. The Δ-causal order defined by Baldoni *et al.* (1998) is correct; however, the use of a global clock is not suitable for distributed communication systems where the message transmission delay is unpredictable and not negligible.

Several existing Δ-causal algorithms (Baldoni *et al.*, 1996; Prakash *et al.*, 1997; Tachikawa and Takizawa, 1997) ensure causal ordering in the presence of lost messages and time delivery constraints; however, in order to achieve this, they all use some type of global reference (shared memory, wall clock, master-slave scheme, etc.).

PRELIMINARIES

The system model

Processes: The application under consideration is composed of a set of processes P = {i, j,...} organized into a group that communicates by passing non reliable broadcast asynchronous messages.

Messages: A finite set of messages M is considered, where, each message m∈M is identified by a tuple m = (p, t), where p∈P is the sender of m, denoted by Src(m) and t is the sequential ordered logical clock for messages of p when m is broadcasted. The set of destinations of a message m is always P.

Events: Let m be a message. The emission event of m by Src(m) is denoted by send(m) and the delivery event of m to participant p ∈ P is denoted by delivery(p, m). The set of events associated to M is then the set E = {send(m): m∈M} ∪ {delivery(p, m) : m ∈ M ∧ p ∈ P}. The process p(e) of an event e∈E is defined by p (send(m)) = p and p(delivery (p, m)) = p. The set of events of a process p is E_p = { e ∈ E: p(e) = p}.

Background and definitions

The happened-before relation: The happened-before relation was defined by Lamport (1978). The happened-before relation establishes possible precedence dependencies in a set of events without using physical clocks. It is a strict partial order (i.e., irreflexive, asymmetric and transitive) defined as follows:

Definition 1: The causal relation → is the least partial order relation on E satisfying the following properties:

- If a and b are events belonging to the same process and a was originated before b, then a→b
- If a is the send message of a process and b is the reception of the same message in another process, then a→b
- If a→b and b→c, then a→c

By using definition 1, a pair of events is said to be concurrently related a || b only if:

$$\neg (a \rightarrow b \vee b \rightarrow a)$$

The precedence relation on messages denoted by m→m' is induced by the precedence relation on events and is defined by:

$$m \rightarrow m' \Leftrightarrow send(m) \rightarrow send(m')$$

The immediate dependency relation: The Immediate Dependency Relation (IDR) introduced by Pomares *et al.* (2004) is the propagation threshold of the control information regarding the messages sent in the causal past that must be transmitted to ensure a causal delivery. The IDR is denoted by ↓ and its formal definition is the following:

Definition 2: Immediate Dependency Relation ↓ (IDR):

$$a \downarrow b \Leftrightarrow [(a \rightarrow b) \wedge \forall c \in E, \neg(a \rightarrow c \rightarrow b)]$$

Thus, an event a directly precedes an event b, iff no other event c belonging to E exists, such that a precedes c and c precedes b.

This relationship is important because if the delivery of messages respects the order of the diffusion for all pairs of messages in an IDR, then the delivery respects the causal order for all messages. This property is formally defined for the broadcast case as follows:

Property 1:

∀ m, m' ∈ M, if send (m) ↓ send (m') ⇒ ∀p ∈ P: delivery (p, m) → deliver (p, m')
then send (m) → send (m') ⇒ ∀p ∈ P: delivery (p, m) → delivery (p, m').

The Δ-causal ordering: The Δ-causal ordering has been introduced in Baldoni *et al.* (1998); it is formally defined for the broadcast case as follows:

Definition 3: A set of events E satisfies the Δ-causal ordering if:

- All events that arrive in Δ are delivered within Δ. All the other events are considered to be lost or discarded and therefore, are never delivered
- All delivery events respect causal ordering, i.e.,

$$\forall m, m' \in M, \text{if send}(m) \rightarrow \text{send}(m'), \text{then } \forall p \in P: \text{delivery}(p, m) \rightarrow \text{delivery}(p, m')$$

The causal distance: The causal distance identifies the number of causal messages that exist in a linearization between a pair of messages in the system (Lopez *et al*., 2005). Formally, the causal distance is defined as follows:

Definition 4: Let m and m' be messages, the distance d (m, m') is defined for any pair m and m' (send(m)→ send(m')), such that d (m, m') is the integer n for some sequence of messages (m_i, i = 0...n), with m = m_0 and m' = m_n, such that send(m_i)↓send(m_{i+1}) for all i = 0…n-1.

In the present study, if more than one linearization between a pair of events exists, the larger causal distance max({d(m, m')}) is taken.

THE Δ-CAUSAL ORDER ALGORITHM

In order to avoid the use of a global clock, an original FEC mechanism is proposed, as well as a distributed lifetime method that verifies if a message satisfies or not its deadline. Here, each mechanism is presented separately and then they are integrated to the minimal broadcast causal algorithm.

The FEC mechanism: All FEC mechanisms introduce some kind of redundancy to support the loss of information. The redundancy in causal algorithms represents the number of times that information about a causal message is sent in the system. The causal algorithm presented in (Pomares *et al*., 2004; Birman, 1993), which uses the IDR relation, is minimal because the IDR relation identifies the necessary and sufficient control information that needs to be sent attached per message. Even when this is a minimal algorithm, redundant control information is still transmitted in some communication scenarios.

The FEC mechanism presented here identifies and uses this inherent redundancy in order to be efficient and will only add extra redundancy when it is needed. The purpose of adding extra redundancy is to increase the probability that causal order delivery will be obtained, even in the presence of lost messages and significant network delays.

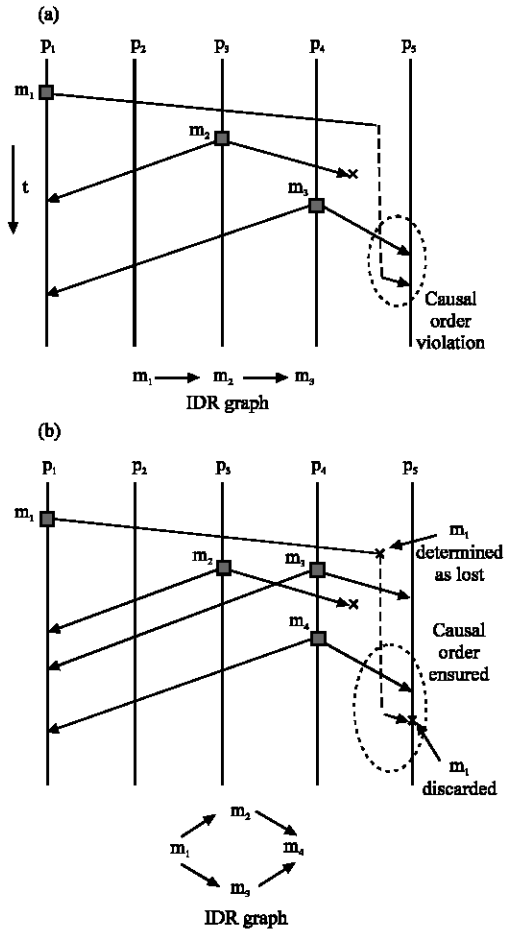

Fig. 1: Example scenarios and their associated IDR graph

Redundancy and the IDR relation: To ensure causal ordering, the minimal algorithm only sends control information attached to each message about messages with an immediate dependency relation. For two messages that are IDR-related (m↓m'), the causal distance is equal to one (d(m, m') = 1). Note that for the serial case, a message m has only one immediate predecessor (best case) and that a message m can have at most n immediate predecessors, one for each process.

For the serial case, for messages that are IDR-related, there is no redundancy in the control information sent. For example, in the serial scenario shown in Fig. 1a, message m_3 only sends information about message m_2 and message m_2 only sends information about message m_1. In this case, if a message is lost, the causal order delivery can be violated. As shown in Fig. 1a, the causal order delivery is violated because at the reception of message m_3, process p_5 cannot determine if a message preceding m_2 exists or not. With the IDR information on m_3, process p_5 can only detect that it missed message m_2. In order not to

stop the system execution, process p_5 considers message m_2 as lost and then delivers m_3. In this scenario, m_1 can be delivered after m_3, which violates the causal ordering.

For the concurrent relation, inherent redundancy exists on the control information sent. For example, in the scenario shown in Fig. 1b, messages m_2 and m_3 have the same immediate predecessor m_1 and therefore, m_2 and m_3 send information about message m_1. If either message m_2 or m_3 is lost, message m_1 can still be detected as shown in Fig. 1b. In this scenario, m_2 is lost and m_3 successfully arrives at p_5. With the IDR information on m_3, process p_5 determines that m_1 exists, which precedes message m_3. To deliver m_3, process p_5 establishes message m_1 as lost. In this scenario, m_1 arrives at p_5 after the delivery of message m_4, but since message m_1 has been established as lost, it is immediately discarded. Therefore, causal order is ensured.

The proposal: In order to support the loss of messages, an increase in the redundancy in the control information sent per message is proposed; this is done by sending information about causally-related messages with a causal distance greater than one. For example, in Fig. 1a, if a causal distance is established to be two (causal_distance = 2), this means that message m_3 must send information about m_2 and m_1.

To be efficient, the redundancy is increased considering the inherent redundancy introduced by the IDR relation. Here, redundancy about a message m, denoted by $redundancy_p(m)$, determines the number of times that the information about a causal message m has been seen (received) by a participaint p. As earlier described, the redundancy increases as the number of concurrent messages increases. Taking into account $redundancy_p(m)$ with a causal distance greater than one (causal_distance > 1) (Appendix 1), one can establish that a message m' must include information about a causal message m ($m \rightarrow m'$) only if the following propagation constraints are satisfied:

PC1: $d(m, m') \leq$ causal_distance and
PC2: causal_distance > $redundancy_p(m)$

With both of these PCs, the control information sent per message is dynamically adapted to the behavior of the system by only introducing redundancy when it is needed. For example, with causal_distance = 2, message m_3, shown in Fig. 1a, must send information about m_2 and m_1, because p_4 has redundancy(m_1) equal to one and a causal distance of $d(m_1, m_3) = 2$ and $d(m_2, m_3) = 1$, respectively. Nevertheless, for the scenario shown in Fig. 1b, message m_4 must send information only about messages m_2 and m_3 and not about m_1, even when $d(m_1, m_4) = 2$. This is done because the redundancy (m_1) seen by p_4 is equal to 2 and therefore, it does not satisfy the second PC.

Note that the value of $redundancy_p(m)$ can differ between participants since it is calculated from the messages received by each one.

The lifetime distributed method: In the transmission of real-time data, such as audio and video, it is possible to establish relative time points (ReTPs) to determine if the data satisfies or not its lifetime. The ReTPs must be dynamically established in order to support random transmission delays.

Establishing relative time points and deadline points: In order to establish the relative time points and deadline points, we assume that the transmission of data, such as audio and video, is executed by transferring messages at a relatively constant rate and that these messages are sequentially timestamped. By taking into account these hypotheses, a ReTP at the reception of the most recent message is established. For example, in Fig. 2, the reception of message m_1 establishes the first relative time point rtp_1; the reception of m_2 establishes the rtp_2 and so on.

Based on the ReTPs, the deadline point of a message m, denoted by deadline (m), is determined from the ReTP previously established. If no message is lost and the messages arrive in order, we have:

$$\text{Deadline}(m_i) = rtp_{i-1} + \Delta: i \geq 1 \quad (1)$$

where, Δ is the message lifetime that establishes the maximum transmission delay supported.

If lost or discarded messages are considered, the equation above is redefined as follows:

$$\text{Deadline}(m_i) = rtp_k + (i-k)\Delta : k < i \text{ and } i \geq 1 \quad (2)$$

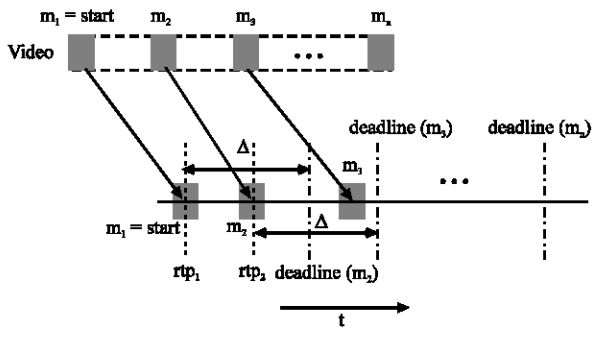

Fig. 2: Streaming scenario

where, rtp_k is the last relative time point established. This general equation is used in the rest of the study to calculate the deadline.

For broadcast asynchronous communication, each process p ∈ P establishes its own ReTPs and therefore, its own deadline points. For this case, a deadline is denoted as deadline (m, p), which determines the deadline point for a message m at a process p.

Data structures: The main data structures used in the algorithm are:

- VT(p) is the vector time. For each process p there is an element VT(p)[j] where j is a process identifier. To refer to a specific process with its respective identifier, p is used when needed. The size of VT is equal to the number of processes in the group. VT(p) contains the local view that process p has of the elements of the system. In particular, element VT(p)[k] represents the greatest element number of the identifier k and seen in causal order by p. It is through the VT(p) structure that the causal delivery of elements can be guaranteed
- CI(p) is the control information structure. It is a set of entries (k, t, d). Each entry in CI(p) denotes a message that satisfies the PC1 and PC2 propagation constraints and that can potentially be attached in the next message m sent by p. The entry (k, t, d) represents a message diffused by participant k at a logical local timeclock t = VT(p)[k] and d contains the redundancy of m = (k, t) seen by p
- The structure of a message m is a quadruplet m = (i, t, content, H(m)), where:

  i is the participant identifier
  t = VT(p)[i] is the logical local clock at node i
  Content is the structure that carries the media data
  H(m) contains the set of all entries (k, t) about messages that satisfy the propagation constraints (PC1 and PC2) with m

- The causal_distance variable is the predetermined causal distance considered
- VTIME (p) is a vector that contains the most recent relative time points. The size of VTIME (p) is equal to VT(p) (one element for each process in the system)
- Current_time (p) represents the physical local time of a process p
- Deadline (k, t) is a function that calculates the deadline point for message m identified by (k,t) where k is the sender process and t is the logical clock

Algorithm description: The Δ-causal algorithm is shown in Fig. 3. When a message m is broadcasted by a process

```
1.  Initially
2.  VT(p)[i] = 0 ∀ i:1…n                    /* Vector clock */
3.  VTIME(p)[i] = 0 ∀ i:1…n                 /* Vector with the ReTPs */
4.  CI(p), H(m), deadline_arr(m) ← Ø
5.  causal_distance = z
6.  let deadline (k, t) = VTIME(p)[k]+(t-VT(p)[k]) *Δ /*Definition of
    deadline (k, t) */
7.  let deadline_arr(m)= {deadline(m)} ∪ {deadline(m') : m' arrived in
    its lifetime and not yet delivered and m→m'}
8.  For each diffusion of message send(m) at p_i
9.   VT(p)[i] = VT(p)[i] +1
10.  for all (k,t,d) ∈ CI(p)
11.    (k, t, d) ← (k, t, d+1)              /* Accounts for redundancy */
12.    H(m) ← H(m) ∪ (k, t) endfor
13.  m = (i, t = VT(p)[i], content, H(m))
14.  Diffusion: send (m)
15.  for all (k, t, d) ∈ CI(p) if d = causal_distance then
16.    CI(p) ← CI(p) / (k,t,d) endfor
17.  For each reception receive(m) at p, m = (k, t, content, H(m))
18.   if (t < VT(p)[k] or deadline(k, t) < current_time(p)) then
19.     If not (t < VT(p)[k]) then
20.             VTIME(p)[k] = current_time(p) endif
21.        VT(p)[k] = max(VT(p)[k], t)
22.        Discard(m)
23.   else
24.    wait ((t = VT(p)[k] + 1 or            / *Δ-causal delivery condition*/
25.    min(deadline(k, VT(p)[k]), {deadline_arr(m)}) < current_time(p)) and
26.    (∀(l,x) ∈ H(m):
27.     x ≤ VT(p)[l] or
28.     min(deadline (l, x), {deadline_arr(m)}) ≤ current_time(p))
29.  Delivery: delivery(m)
30.  VTIME(p)[k] = current_time(p)
31.  VT(p)[k] = max(VT(p)[k], t)
32.  for all (l,x) ∈ H(m) if x >VT(p)[l]      /* For missing messages */
33.       VT(p)[l]=x endfor
34.  if (∃(l, x, d) ∈ CI(p) | l = k ) then    /* Keeps the most
35.       CI(p) ← CI(p) / (l, x, d) endif     recent message
36.  CI(p) ← CI(p) ∪ {(k, t, d = 0)}          sent by p_k */
37.  for all (l, x) ∈ H(m) if ∃d : (l, x, d) ∈ CI(p) then
38.     (l, x, d) ← (l, x, d+1) endfor /* Accounts for redundancy*/
39.     for all (l, x, d) ∈ CI(p) if d=causal_distance then
40.        CI(p) ← CI(p)/ (l, x, d) endfor
41.   endif
```

Fig. 3: Broadcast Δ-causal algorithm

p, the H(m) is constructed by adding entries from the CI(p) (lines 10-12) to it. Each entry in H(m) satisfies, with to respect to m, the PC1 and PC2 propagation constraints. In order to comply with PC1 and PC2, we use a logical counter d by each entry in CI(p) ((k, t, d) ∈ CI(p)). The variable d is increased by one each time that its associated entry is added to a H(m) by a process p (line 11) or when that entry is received in a H(m) (lines 37-38). The variable d contains the redundancy of the message m = (k, t) seen by p. Note that only when no concurrent messages exist, does the value of d specify the causal distance between events.

The Δ-causal delivery condition: At the reception of a message, m = (k, t, content, H(m)) will be immediately discarded if it has already been marked as lost (t<VT(p)[k]) or if it misses its deadline (line 18). If m is not discarded, it is delivered as soon as the Δ-causal delivery condition becomes true (lines 24-28). This delivery

condition ensures that a message m is delivered in its lifetime (lines 25 and 28) and that it will be delivered if and only if all messages causally related to it have either been delivered or have been established as missing. A posteriori, these messages are marked as lost in lines 32-33 and therefore, they will never be delivered.

Overhead analysis: In order to be efficient, each entry in CI(p) and eventually in H(m) corresponds to the most recent message sent by a process $p_i \in P$ and causally received by p. This is possible since each message m is sequentially timestamped with its respective local logical clock of p_i = Src(m). By knowing the sequential order, a participant p_j can determine at any message reception if a message or set of messages diffused by p_i has been lost, independently of the causal distance.

Since, H(m) only has the most recent messages that precede a message m, the overhead per message in this algorithm to ensure Δ-causal ordering is given by the cardinality of H(m), which can fluctuate in the case between 0 and n-1 ($0 \leq |H(m)| \leq n-1$). For the serial case, |H(m)| is at most the causal_distance established ($|H(m)| \leq$ causal_distance) and for the case of concurrent messages, the worst case is at most n-1 ($|H(m)| \leq n-1$), which is the same boundary for messages that are IDR related (causal_distance = 1). Note that in the algorithm proposed in this article, as for the minimal causal algorithm in (Pomares *et al*., 2004), the likelihood that the worst case will occur approaches zero as the number of participants in the group grows. The worst case of the algorithm presented here is the constant overhead attached per message by algorithms that are exclusively based on vector clocks (Mattern, 1989).

CORRECTNESS PROOF

To show that the algorithm presented here ensures the Δ-causal delivery (correctness), a sketch of proof is given as follows. In the proof, the referenced lines correspond to the lines of the earlier Fig. 3.

Theorem 1: (Liveness) (i) all messages arriving within their deadlines and whose deliveries do not violate causal ordering will be delivered within their deadlines and (ii) all messages arriving after the expiration of their deadlines or whose delivery would cause a causal violation will be discarded.

Proof: Point (ii) is ensured from the test of line 18. Point (i) is proved by contradiction. Suppose that there exists a message m = (k, t) that arrived within its deadline and has not been delivered within its deadline. To proof this, lemma 1 is introduced.

Lemma 1: Each variable in VTIME(p)[i], for all i:1…n does not decrease.

The proof follows directly from the algorithm (lines 18 and 29)

To proof point (i) two cases exist:

Messages from the same source: For this case, by using lemma 1, the following property is presented:

(P1): For all $m = (k, t) \in M$: $Src(m) = p_k \Rightarrow \forall\ p \in P$, $deadline_p(k, t') < deadline_p(k, t)\ \forall\ t'$: 1, 2,…,t−1

Denying the first part of the delivery condition (line 25) that corresponds to messages from the same source, it follows that

$$\exists\ (k, t'), t' < t : (deadline(k, t' = VT(p)[k])) \geq current\_time(p))$$

On the deadline of message m = (k, t), one has that current_time(p) = deadline (k, t). So by direct replacement, it follows that:

$$\exists\ (k, t'), t' < t : (deadline\ (k, t') \geq deadline\ (k, t))$$

This sentence contradicts property P1. It follows that at the deadline of an arrived message m, the first part of the denied delivery condition is false, thus contradicting the initial assumption.

Messages from a different source: To proof this case, two functions need to be first introduced: $delivery\_time_p(m)$, which is the time when a message m is delivered at a process p and $discarded\_time_p(m)$, which is the time when a message m is marked as missing at a process p. By using lines 25, 28 and Lemma 1, the second and third properties are as follows:

(P2): For all m', $m \in M$, $m' \rightarrow m$ received at $p \in P \Rightarrow delivery\_time_p(m') \leq deadline_p(m)$

(P3): For all m', $m \in M$, $m' \rightarrow m$: m' has not been received at $p \in P \Rightarrow discarded\_time_p(m') \leq deadline_p(m)$

Next, the proof that involves P2 is presented. The proof involving P3 is similar and not presented here.

For $m' \rightarrow m$ received at $p \in P$. By denying the second part of the delivery condition (line 28), it follows that:

$$\exists m' = (l,x) \in H(m):$$
$$(min(deadline\ (l, x), \{deadline\_arr(m)\}) > current\_time(p))$$

If $delivery\_time_p(l, x) = min\ (deadline\ (l, x), \{deadline\_arr(m)\})$, it follows that:

$\exists$ m' = (l,x)$\in$H(m):
(delivery_time$_p$(l,x)) > current_time(p))

On the deadline of message m = (k, t), current_time(p) = deadline(m). So by direct replacement, it follows that:

$\exists$ m' = (l, x)$\in$H(m):
(delivery_time$_p$(m')) > deadline(m))

This sentence contradicts property P2. It follows that at the deadline of an arrived message m, the second part of the denied delivery condition is false, thus contradicting the initial assumption. □

Lemma 2: For all m', m $\in$ **M**, m' $\rightarrow$ m such that Src(m')$\neq$Src(m) and redundancy(m') $\leq$ causal_distance implies that m' = (l, x) $\in$ H(m)

This is accomplished by the procedures at the diffusion message by lines 10 and 15 and at the reception message by lines 36, 37 and 39.

Theorem 2: (Correctness) for all m', m $\in$ **M**, m'$\rightarrow$ m such that d (m', m) $\leq$ causal_distance implies that delivery (m') $\rightarrow$ delivery (m).

Proof: Consider two messages m_0 and m_n such that send(m_0) $\rightarrow$ send(m_n) and both are received by p. The fact that they are delivered to p according to causal ordering is shown.

For this proof, there are two general cases. The proof is by induction on the distance d (m_0,m_n).

Base case: d (m_0,m_n) = 1 and d (m_0,m_n) $\leq$ causal_distance In this case, m_0 is IDR related to m_n and from lemma 2 and since always d (m', m) $\leq$ redundancy(m'), then $m_0 \in$ H(m_n). It follows that line 27 will delay the delivery of m_n until after the delivery of m_0.

Induction case: d(m_0,m_n)$\geq$2 and d(m_0, m_n)$\leq$causal_distance By induction, all messages of the set {$m_i \in$ **M**: $m_{i-1} \downarrow m_i$ for all i = 1…n−1} that are delivered to p are delivered in causal order. For the induction phase, there are two cases depending on whether m_{n-1} has been delivered or discarded at p.

For m_{n-1} delivered at p: We have m_{n-1} that immediately precedes m_n so the base case applies to these messages: m_{n-1} is delivered before m_n and by transitivity m_0 is delivered before m_n.

For m_{n-1} discarded at p: In this case, $m_{n-1} \in$ H(m_n) and by lemma 1 and 2 and P3, it follows that m_n is delivered after that discarded_time$_p$(m_{n-1}). By lines 25, 28 and lemma 1, for a message m_i that belongs to the path m_0 to m_{n-1} implies that the delivery or discarded time of m_i is less than or equal to the discarded time of m_{n-1}. Consequently, m_n is delivered at p after m_0. □

Note that when a message m_{n-x} such that n−x > causal_distance, then $m_{n-x} \notin$ H(m_n) and therefore, the causal delivery of m_{n-x} with respect to m_n cannot be ensured.

CONCLUSION

An efficient Δ-causal algorithm has been presented. The algorithm is efficient since the control information attached per message is dynamically adapted to the behavior of the system. This control information allows a causal forward error recovery when messages are lost. The algorithm presented ensures Δ-causal order delivery without using a global clock. To avoid the use of a global clock, an original FEC mechanism and a distributed lifetime method is proposed. The Δ-causal algorithm presented is suitable for real-time distributed systems since it performs a forward error recovery and it neither uses global references nor requires previous knowledge of the behavior of the system.

APPENDIX 1

Analysis of probabilities: Consider E_i independent events (send events) with i = 1,…..m and suppose that the rate for their delivery or loss is λ>0, obeying a Poisson distribution.

$$p(j) = p\{X = j\} = e^{-\lambda}\frac{\lambda^j}{j!}, \qquad j = 0,1,.....,m,$$

where, X is a random variable that takes one of the values 0,1……

The events are considered to be successful if they arrive to their destination and unsuccessful if they do not arrive. Suppose that there are n<m events of the m possible ones, then

- The probability that an event is unsuccessful is:

$$p(0) = e^{-\lambda}$$

- The probability that at least one event is unsuccessful is given by:

$$p\{X \geq 1\} = 1 - p(0) = 1 - e^{-\lambda}$$

- The probability that there is no more than n successful events is given by:

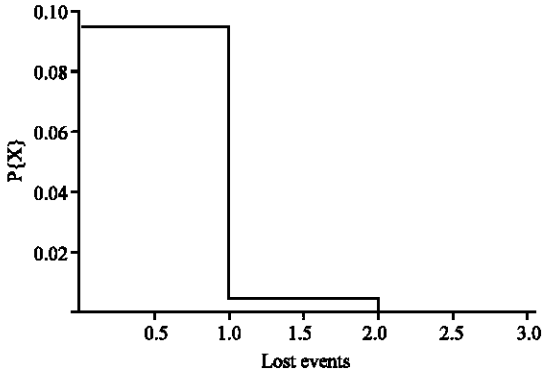

Fig. 4: Probability that at least n events or more are unsuccessful.

$$p\{X \le n\} = e^{-\lambda}\sum_{i=0}^{n}\frac{\lambda^i}{i!}$$

- The probability that there are at least n or more unsuccessful events that do not arrive, is given by:

$$p\{X \ge n\} = 1 - e^{\lambda}\sum_{i=0}^{n}\frac{\lambda^i}{i!}$$

If one considers a loss rate of $\lambda = 0.1$ (Olsen, 2003), the diagram corresponding to case 4, which is the case of interest in this study is shown in Fig. 4.

As can be shown in Fig. 4, the diagram approaches zero rather rapidly. From values n = 3, the likelihood that at least n events or more are unsuccessful becomes negligible.

REFERENCES

Baldoni, R., M. Raynal, R. Prakash and M. Singhal, 1996. Broadcast with time and causality constraints for multimedia applications. Proceedings of the 22nd EUROMICRO Conference on Hardware and Software Design Strategies, September 2-5, Prague, Czech Republication, pp: 617-624.

Baldoni, R., R. Prakash, M. Raynal and M. Singhal, 1998. Efficient Δ-causal broadcasting. Int. J. Comput. Syst. Sci. Eng., 13: 263-269.

Birman, K., 1993. The process group approach to reliable distributed computing. Commun. ACM, 36: 37-53.

Lamport, L., 1978. Time clocks and the ordering of messages in distributed systems. Commun. ACM, 21: 558-565.

Lopez, E., J. Estudillo, J. Fanchon and S. Pomares, 2005. A fault-tolerant causal broadcast algorithm to be applied to unreliable networks. Proceeding of 17th International Conference on Parallel and Distributed Computing and Systems, Nov. 14-16, Phoenix, AZ. USA., pp: 465-470.

Lopez, E., S. Pomares and G. Rodriguez, 2008. MOCAVI: An efficient causal protocol to cellular networks. Int. J. Comput. Sci. Network Security, 8: 136-144.

Mattern, F., 1989. Virtual time and global states of distributed systems. Proceedings of the International Workshop on Parallel and Distributed Algorithms, 1989, Department of Computer Science, pp: 215-226.

Nishimura, T., N. Hayashibara, T. Enokido and M. Takizawa, 2005. Causally ordered delivery with global clock in hierarchical group. Proceedings of 11th International Conference on Parallel and Distributed Systems, July 20-22, IEEE Xplore London, pp: 560-564.

Olsen, J., 2003. Stochastic Modeling and Simulation of the TCP Protocol. 1st Edn., Uppsala University, Uppsala, Sweden.

Perkins, C., 2003. RTP Audio and Video for Internet. 1st Edn., Addison Wesley, USA., ISBN: 0-672-32249-8.

Plesca, C., R. Grigoras, P. Queinnec, G. Padiou and J. Fanchon, 2006. A coordination-level middleware for supporting flexible consistency in CSCW. Proceedings of the 14th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, Feb. 15-17, IEEE Xplore London, pp: 316-321.

Pomares, S., J. Fanchon and K. Drira, 2004. The inmediate dependency relation: An optimal way to ensure causal group communication. Ann. Rev. Scalable Comput., 6: 61-79.

Prakash, R., M. Raynal and M. Singhal, 1997. An adaptive causal ordering algorithm suited to mobile computing environment. J. Parallel Distrib. Comput., 41: 190-204.

Tachikawa, T. and M. Takizawa, 1997. &Delta-causality in wide-area group communications. Proceedings of International Conference on Parallel and Distributed Systems, Dec. 10-13, Seoul, South Korea, pp: 260-267.